Математическая статистика

Практическое задание 4

В данном задании предлагается провести некоторое исследование доверительных интервалов и байесовких методов.

Правила:

- Выполненную работу нужно отправить на почту `probability.diht@yandex.ru`, указав тему письма `"[номер группы] Фамилия Имя - Задание 4"`. Квадратные скобки обязательны. Вместо `Фамилия Имя` нужно подставить свои фамилию и имя.
- Прислать нужно ноутбук и его pdf-версию. Названия файлов должны быть такими: `4.N.ipynb` и `4.N.pdf`, где `N` - ваш номер из таблицы с оценками.
- Никакой код из данного задания при проверке запускаться не будет.
- Некоторые задачи отмечены символом *. Эти задачи являются дополнительными. Успешное выполнение большей части таких задач (за все задания) является необходимым условием получения бонусного балла за практическую часть курса.
- Баллы за каждую задачу указаны далее. Если сумма баллов за задание меньше 25% (без учета доп. задач), то все задание оценивается в 0 баллов.

Баллы за задание:

- Задача 1 - 5 баллов
- Задача 2* - 3 балла
- Задача 3* - 3 балла
- Задача 4 - 5 баллов
- Задача 5* - 2 балла
- Задача 6 - 4 балла
- Задача 7 - 1 балл
- Задача 8 - 3 балла
- Задача 9* - 5 баллов
- Задача 10 - 5 баллов
- Задача 11* - 3 балла

In [52]:

```
import numpy as np
import scipy.stats as sps
import matplotlib.pyplot as plt
import pandas as pd

%matplotlib inline
```

1. Доверительные интервалы

Задача 1. В этой задаче нужно визуализировать доверительные интервалы для выборок из различных распределений. Чтобы не плодить код, напишите следующую функцию. Пример построения есть в ячейке №22 ноутбука `python_6`.

In [53]:

```
def draw_confidence_interval(left,  # левая граница интервала
                             right,  # правая граница интервала
                             estimation=None,  # если задана, то рисуется график оценки
                             sample=None,  # если задано, то рисуются точки выборки
                             ylim=(None, None)):  # ограничение по оси y
    length = len(sample)
    x = np.arange(1, length+1)

    plt.figure(figsize=(15, 8))
    if(sample is not None):
        plt.scatter(x, sample, alpha=0.4, s=40, label='sample')
    if(estimation is not None):
        plt.plot(x, estimation, color='red', linewidth=2.5, label='estimation')  # linewidt
    # заполняет пространство между двумя функциями
    plt.fill_between(x, right, left, alpha=0.3, label='confidence interval')
    plt.legend()
    plt.ylim(ylim)
    plt.grid(ls=':')
    plt.show()
```

Сгенерируйте выборки и постройте графики доверительных интервалов по следующей схеме.

- Выборка из распределения $\mathcal{N}(0, 1)$; точный доверительный интервал минимальной длины в модели $\mathcal{N}(\theta, 1)$; нужно нанести на график точки выборки.
- Выборка из распределения $U[0, 1]$; точный доверительный интервал минимальной длины в модели $U[0, \theta]$ на основе статистики $X_{(n)}$; нужно нанести на график точки выборки.
- Выборка из распределения $\Gamma(3, 2)$; точный асимптотический доверительный интервал в модели $\Gamma(\theta, 2)$; точки выборки наносить на график не нужно.
- Выборка из стандартного распределения Коши; точный асимптотический доверительный интервал в модели распределения Коши со сдвигом; нужно нанести на график точки выборки.
- Выборка из стандартного распределения Коши; точный доверительный интервал минимальной длины в модели $\mathcal{N}(\theta, 1)$; нужно нанести на график точки выборки.

Генерировать выборки размера 100, уровень доверия брать $\alpha = 0.95$. Для вычисления квантилей у каждого распределения из `scipy.stats` есть функция `ppf`.

Сделайте вывод. Насколько часто истинное значение параметра попадает в доверительный интервал? Как длина интервала зависит от размера выборки?

In [4]:

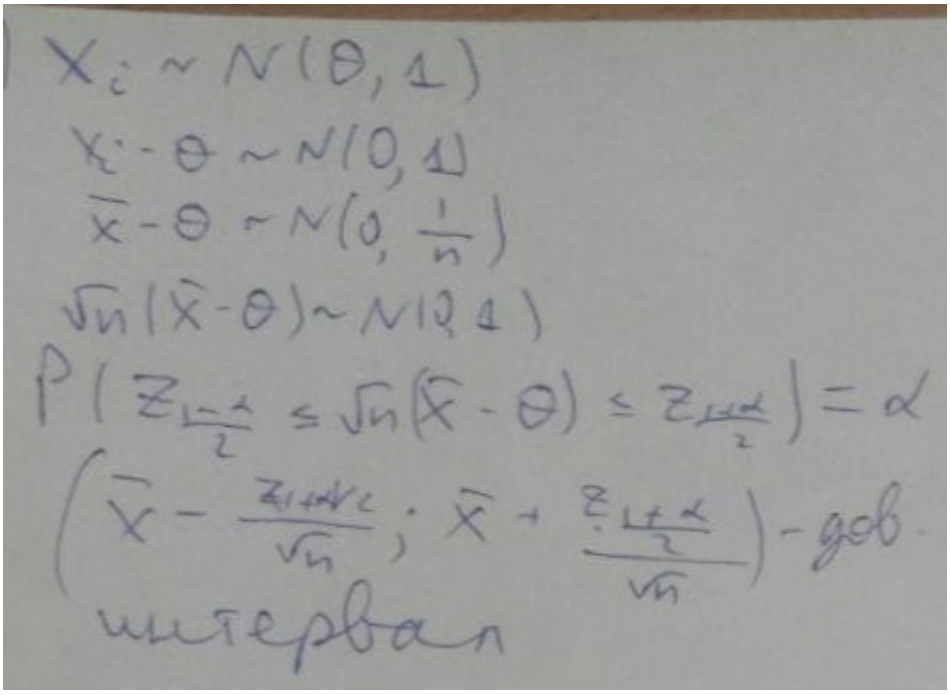
```
from IPython.display import Image
```

Выборка из распределения $\mathcal{N}(0, 1)$; точный доверительный интервал минимальной длины в модели $\mathcal{N}(\theta, 1)$; нужно нанести на график точки выборки

In [5]:

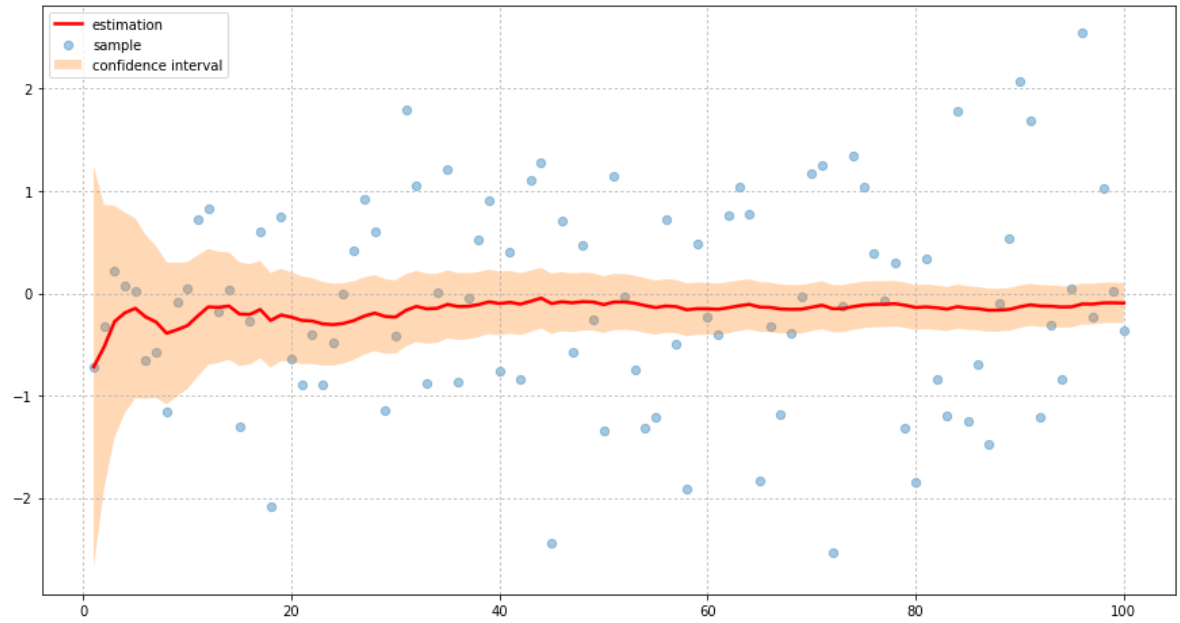
```
Image(filename='task1_1.jpg')
```

Out[5]:



In [56]:

```
alpha=0.95
sample = sps.norm(loc=0, scale=1).rvs(size=100)
cummean = np.cumsum(sample) / np.arange(1, 101)
left = cummean - sps.norm.ppf((1 + alpha) / 2) / np.sqrt(np.arange(1, 101))
right = cummean + sps.norm.ppf((1 + alpha) / 2) / np.sqrt(np.arange(1, 101))
draw_confidence_interval(left, right, estimation=cummean, sample=sample)
```

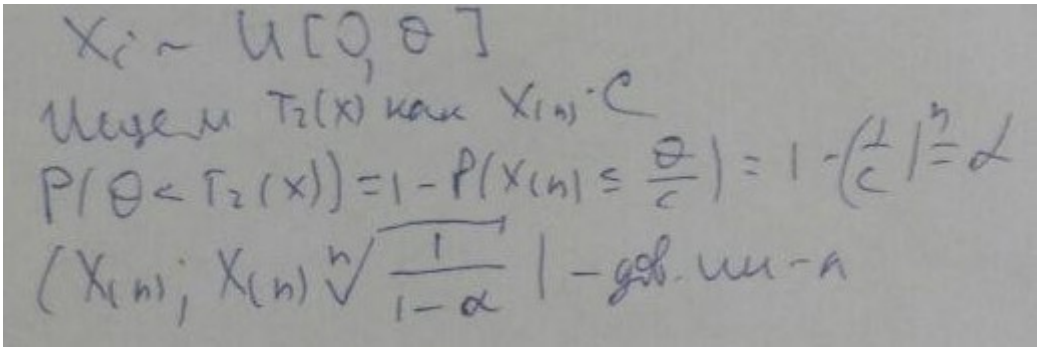


Выборка из распределения $U[0, 1]$; точный доверительный интервал минимальной длины в модели $U[0, \theta]$ на основе статистики $X_{(n)}$; нужно нанести на график точки выборки

In [7]:

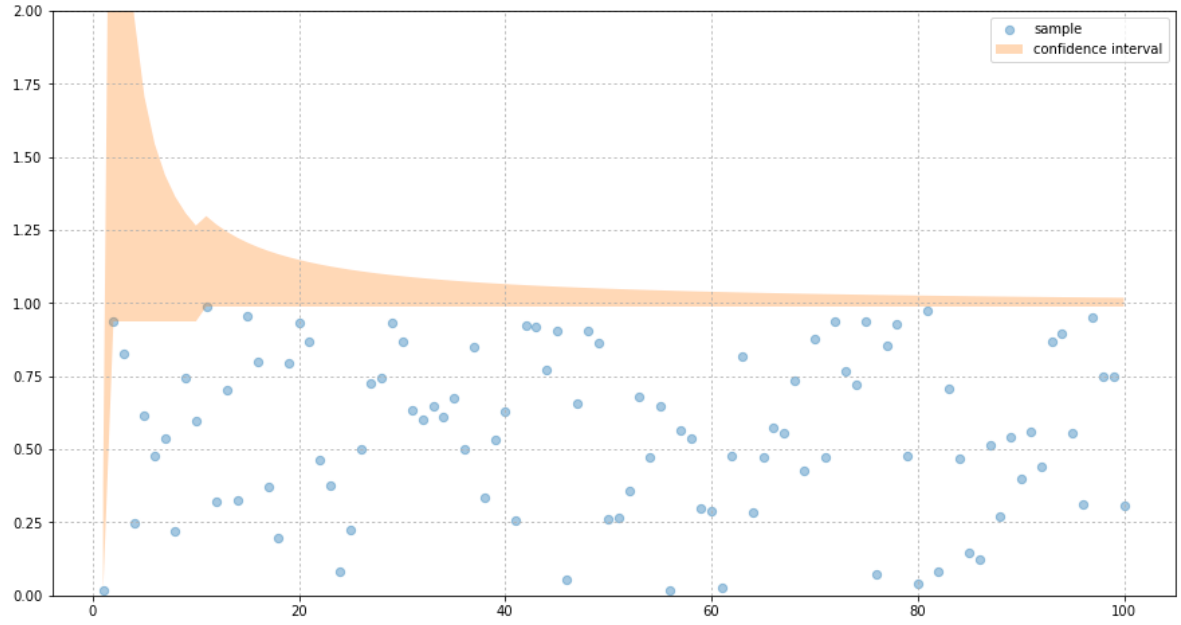
```
Image(filename='task1_2.jpg')
```

Out[7]:



In [8]:

```
sample = sps.uniform(loc=0, scale=1).rvs(size=100)
left = np.maximum.accumulate(sample)
right = left / (1 - alpha) ** (1 / np.arange(1, 101))
draw_confidence_interval(left=left, right=right, sample=sample, ylim=(0, 2))
```

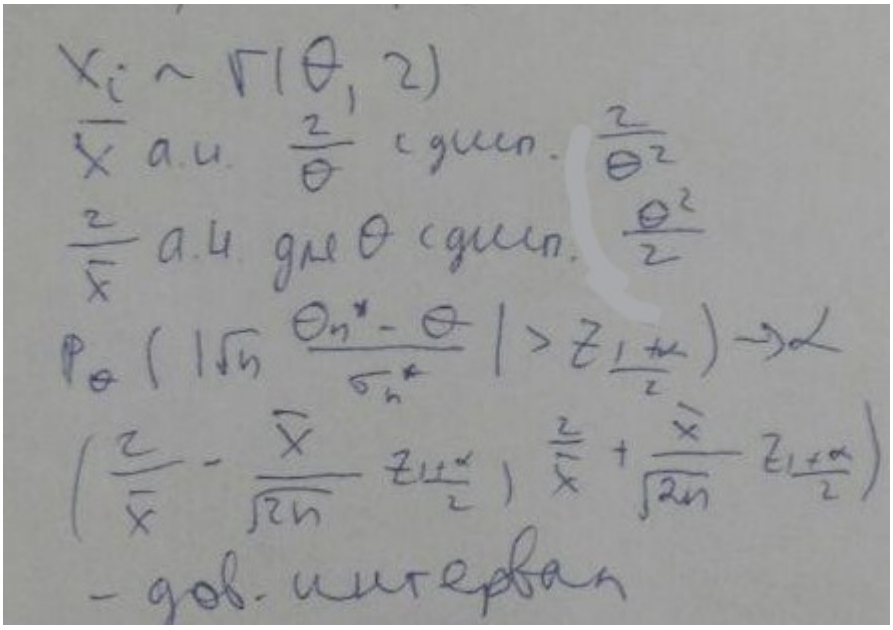


Выборка из распределения $\Gamma(3,2)$; точный асимптотический доверительный интервал в модели $\Gamma(\theta,2)$; точки выборки наносить на график не нужно

In [9]:

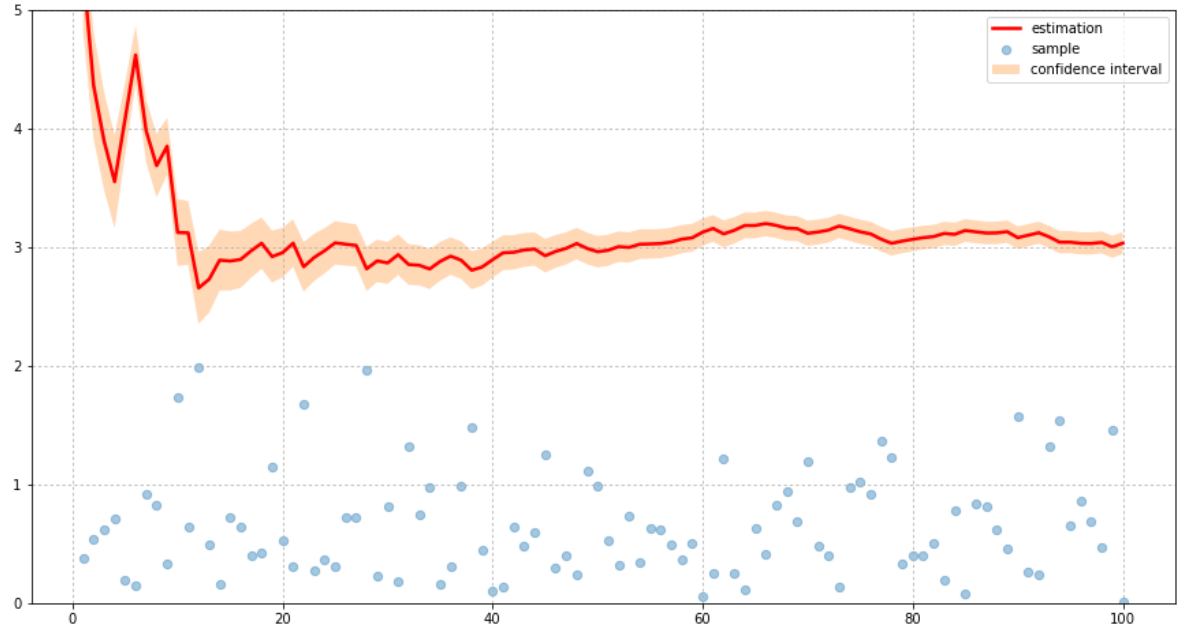
```
Image(filename='task1_3.jpg')
```

Out[9]:



In [57]:

```
sample = sps.gamma(a=2, scale=1/3).rvs(size=100)
cummean = np.cumsum(sample) / np.arange(1, 101)
left = 2 / cummean - cummean / np.sqrt(2 * np.arange(1, 101))  * sps.norm.ppf((1 + alpha) /
right = 2 / cummean + cummean / np.sqrt(2 * np.arange(1, 101))  * sps.norm.ppf((1 + alpha)
draw_confidence_interval(left=left, right=right, estimation=2 / cummean, sample=sample, yli
```

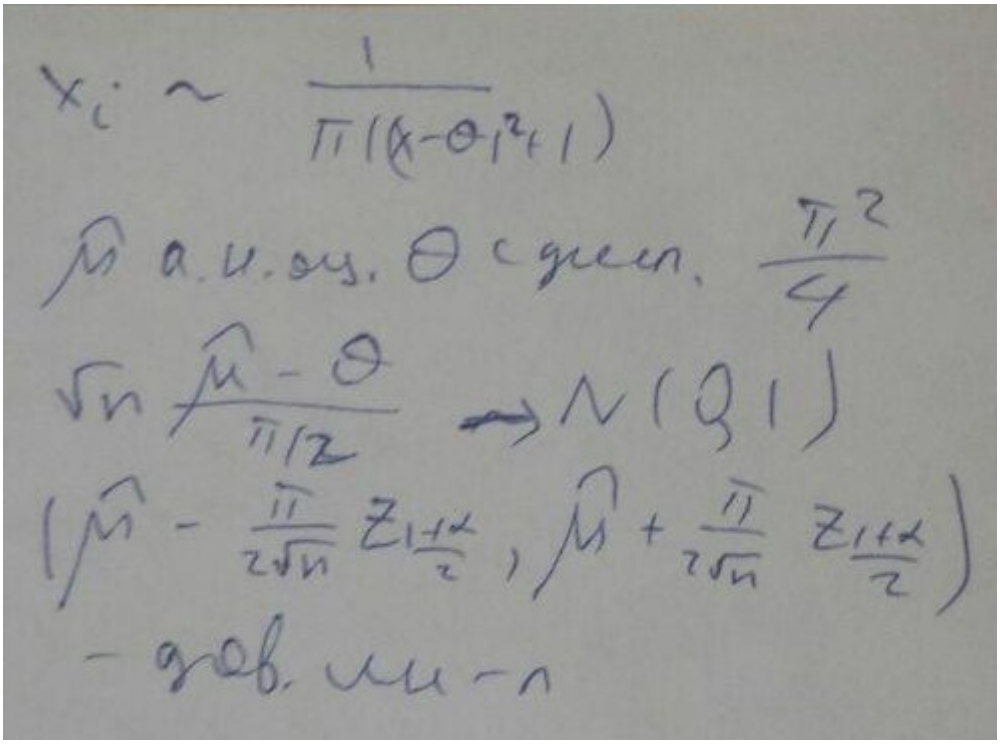


Выборка из стандартного распределения Коши; точный асимптотический доверительный интервал в модели распределения Коши со сдвигом; нужно нанести на график точки выборки

In [13]:

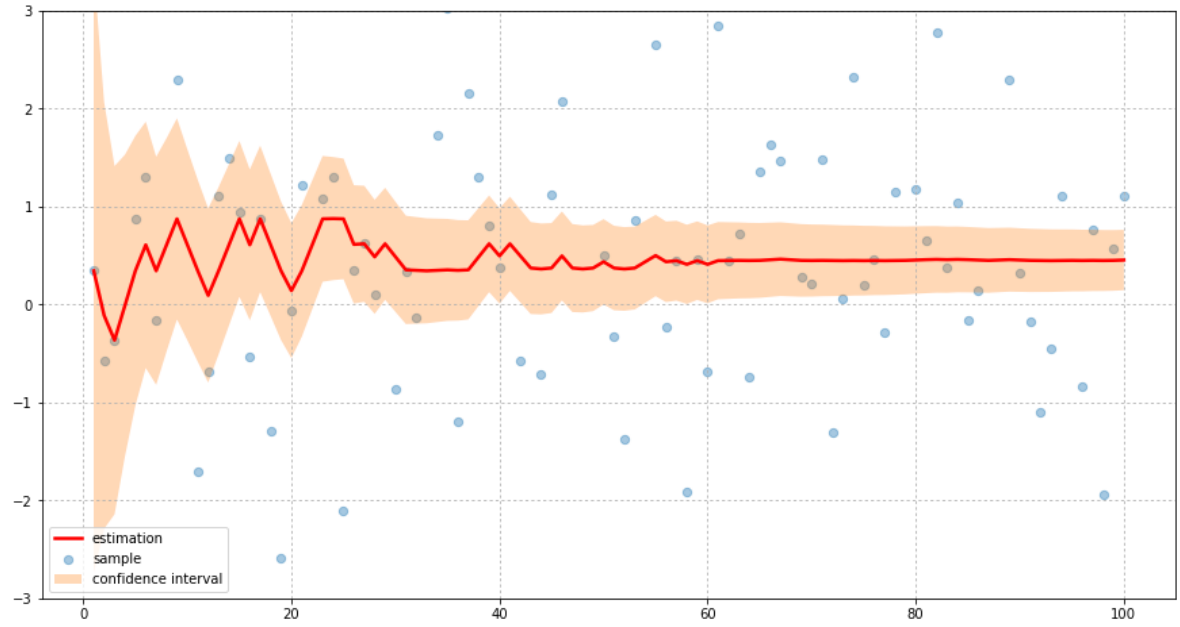
```
Image(filename='task1_4.jpg')
```

Out[13]:



In [58]:

```
sample = sps.cauchy.rvs(size=100)
cummedian = list(map(lambda n:np.median(sample[:n]), np.arange(1, 101)))
left = cummedian - np.pi / np.sqrt(4 * np.arange(1, 101))  * sps.norm.ppf((1 + alpha) / 2)
right = cummedian + np.pi / np.sqrt(4 * np.arange(1, 101))  * sps.norm.ppf((1 + alpha) / 2)
draw_confidence_interval(left=left, right=right, estimation=cummedian, sample=sample, ylim=
```

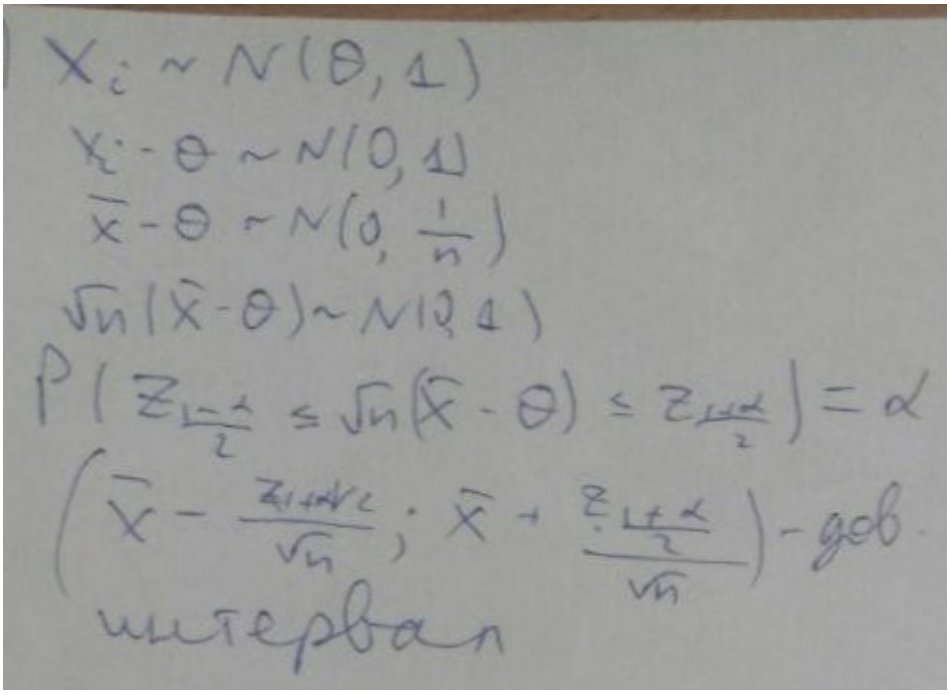


Выборка из стандартного распределения Коши; точный доверительный интервал минимальной длины в модели $\mathcal{N}(\theta, 1)$; нужно нанести на график точки выборки

In [15]:

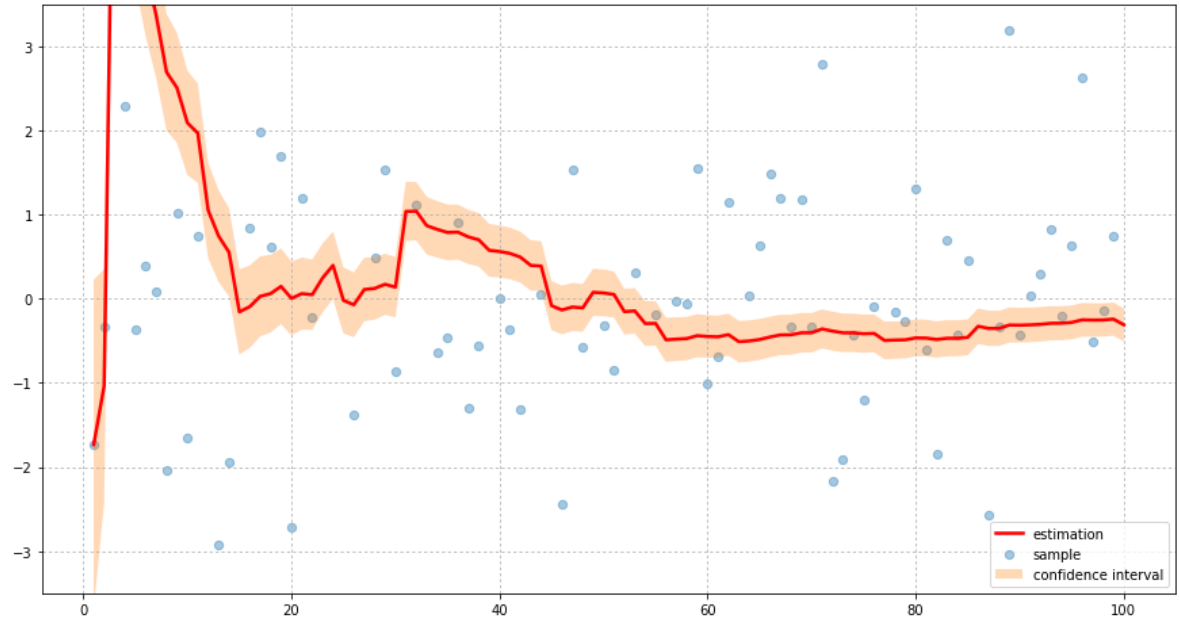
```
Image(filename='task1_1.jpg')
```

Out[15]:



In [60]:

```
sample = sps.cauchy.rvs(size=100)
cummean = np.cumsum(sample) / np.arange(1, 101)
left = cummean - sps.norm.ppf((1 + alpha) / 2) / np.sqrt(np.arange(1, 101))
right = cummean + sps.norm.ppf((1 + alpha) / 2) / np.sqrt(np.arange(1, 101))
draw_confidence_interval(left, right, estimation=cummean, sample=sample, ylim=(-3.5, 3.5))
```



Вывод

1)Видим, что в первых 4 случаях, когда модель была подобрана правильно, значение параметра попадает в интервал. Причём при увеличении выборки длина интервала уменьшается

2)Но видно, что при размере выборки 40 длина интервала незначительно больше длины интервала при размере 100. Т.е. для определения параметра достаточно иметь выборку размера 40

3)В последнем примере модель была выбрана неправиьная, поэтому значение параметра не определить(например мы получили значение 0, хотя должно быть 1)

Задача 2*. Аналогично заданию 1 постройте доверительные интервалы для следующих случаев

- Выборка из распределения $\Gamma(3, 2)$; точный асимптотический доверительный интервал для θ в модели $\Gamma(\theta, \beta)$, причем β неизвестно; точки выборки наносить на график не нужно. Сравните с интервалом для случая известного β.
- Выборка из распределения $\Gamma(3, 2)$; точный асимптотический доверительный интервал для β в модели $\Gamma(\theta, \beta)$, причем θ неизвестно; точки выборки наносить на график не нужно.

Задача 3*. Сгенерируйте выборку размера 200 из распределения $\mathcal{N}((0, 0)^T, ((2, 1)^T, (1, 3)^T))$. Постройте точную доверительную область для θ в модели $\mathcal{N}(\theta, ((2, 1)^T, (1, 3)^T))$. Нанесите на график точки выборки.

Задача 4. При использовании асимптотических доверительных интервалов важно понимать, какой размер выборки является достаточным для хорошего приближения. Иначе говоря, пусть $\xi_n \xrightarrow{d} \mathcal{N}(0, 1)$. Начиная с какого n распределение статистики ξ_n хорошо приближается нормальным распределением?

Для ответа на этот вопрос проведите следующее исследование. Сгенерируйте $K = 10^5$ выборок $(X_{i,k},\ i \leqslant N)$ размера $N = 300$, где $k \leqslant K$ --- номер выборки. Для каждой подвыборки k-ой выборки $(X_{i,k},\ i \leqslant n)$ посчитайте значение статистики $T_{n,k}$ (определение далее) для всех $n \leqslant N$. Далее для каждого фиксированного n постройте эмпирическую функцию распределения F_n^* по выборке $(T_{n,k}, k \leqslant K)$ и посчитайте точное значение статистики $D_n = \sup\limits_{x \in \mathbb{R}} |F_n^*(x) - F(x)|$, где F --- функция распределения $\mathcal{N}(0, 1)$ (см. задачу 4 задания 1). Постройте график зависимости D_n от n.

Рассмотрите следующие случаи

- $X_1, \ldots, X_n \sim \mathcal{N}(0, 1)$. Рассмотреть $T = \sqrt{n} \cdot \overline{X}$ и $T = \sqrt{n} \cdot \overline{X} \Big/ \sqrt{S^2}$.
- $X_1, \ldots, X_n \sim Bern(p), p = 0.5$. Рассмотреть $T = \sqrt{n}\frac{\overline{X} - p}{\sqrt{p(1-p)}}$ и $T_n = \sqrt{n}\frac{\overline{X} - p}{\sqrt{S^2}}$.
- $X_1, \ldots, X_n \sim Cauchy$. Рассмотреть $T = \sqrt{n}\frac{\widehat{\mu}}{\pi/2}$.

В первых двух пунктах нужно построить две зависимости на одном графике для сравнения. Масштаб графика должен быть таким, чтобы четко можно было увидеть различие между двумя статистиками. Например, поставьте ограничение сверху по оси y на 0.05. Незабудьте добавить сетку и легенду.

Старайтесь не копировать много кода, пишите вспомогательные функции. Обратите внимание, что оптимальный код для первых двух пунктов выполняется за 30 секунд, для третьего --- за 3 минуты. Неоптимальный код может выполняться более часа.

Сделайте вывод о том, в каком случае распределение статистики быстрее приближается нормальным распределением. Начиная с какого размера выборки можно пользоваться приближением нормальным распределением?

In [ ]:

Задача 5*. Проведите исследование аналогичное задаче 4 для статистик из задачи 2.

Задача 6. Реализуйте следующую функцию для выборки из нормального распределения

In [38]:

```
def normal_summary(sample):
    size = len(sample)
    sample_mean = np.mean(sample)
    sample_median = np.median(sample)
    sample_std = np.std(sample)
    left_confidence = sample_mean - sps.norm.ppf((1 + alpha) / 2) * sample_std / np.sqrt(si
    right_confidence = sample_mean + sps.norm.ppf((1 + alpha) / 2) * sample_std / np.sqrt(s
    kstest = sps.kstest(sample, sps.norm(loc=sample_mean, scale=sample_std).cdf)
#     print('size: %d' % size)
#     print('sample mean: %.2f' % sample_mean)
#     print('sample median: %.2f' % sample_median)
#     print('sample std: %.2f' % sample_std)  # стандартное отклонение = корень из дисперси
#     print('0.95 confidence interval: (%.2f, %.2f)' % (left_confidence, right_confidence))
#     print('KS-stat: %.3f' % kstest.statistic)  # значение статистики из теоремы Колмогоро
#                                     # взяв в качестве F функцию распределения нормального
#                                     # распределения с оцененными выше параметрами
    return size, sample_mean, sample_median, sample_std, (left_confidence, right_confidence
```

In [39]:

```
normal_summary(sps.norm(loc=2, scale=5).rvs(size=1000))
```

Out[39]:

```
(1000,
 2.0221089994328598,
 2.0855040835077063,
 5.1561752042766793,
 (1.7025318217025953, 2.3416861771631243),
 KstestResult(statistic=0.02501219640999286, pvalue=0.55892641482670702))
```

Протестируйте функцию на выборках из нормального распределения и на выборках из других распределений. Какой вывод можно сделать о поведении статистики Колмогорова-Смирнова?

In [51]:

```
sizes, means, medians, stds, confidence_intervals, kstests = [], [], [], [], [], []

norm_sample_0 = sps.norm.rvs(size=10)
norm_sample_1 = sps.norm.rvs(size=100)
norm_sample_2 = sps.norm.rvs(size=1000)
norm_sample_3 = sps.norm.rvs(size=10000)
pareto_sample_0 = sps.pareto(b=3).rvs(size=10)
pareto_sample_1 = sps.pareto(b=3).rvs(size=100)
pareto_sample_2 = sps.pareto(b=3).rvs(size=1000)
pareto_sample_3 = sps.pareto(b=3).rvs(size=10000)
uniform_sample_0 = sps.uniform.rvs(size=10)
uniform_sample_1 = sps.uniform.rvs(size=100)
uniform_sample_2 = sps.uniform.rvs(size=1000)
uniform_sample_3 = sps.uniform.rvs(size=10000)

samples = [norm_sample_0, norm_sample_1, norm_sample_2, norm_sample_3,
           pareto_sample_0, pareto_sample_1, pareto_sample_2, pareto_sample_3,
           uniform_sample_0, uniform_sample_1, uniform_sample_2, uniform_sample_3,]

for sample in samples:
    size, sample_mean, sample_median, sample_std, confidence_interval, kstest = normal_summ
    sizes.append(size)
    means.append("{:.2}".format(sample_mean))
    medians.append("{:.2}".format(sample_median))
    stds.append("{:.2}".format(sample_std))
    confidence_intervals.append("({:.2}, {:.2})".format(confidence_interval[0], confidence_
    kstests.append("({:.2},{:.2})".format(kstest[0], kstest[1]))

table  = pd.DataFrame({
        "Type" : ['Normal', 'Normal', 'Normal', 'Normal',
                  'Pareto', 'Pareto', 'Pareto', 'Pareto',
                  'Uniform', 'Uniform', 'Uniform', 'Unifrom'],
        "Size" : sizes,
        "Mean" : means,
        "Median" : medians,
        "Std" : stds,
        "Confidence interval" : confidence_intervals,
        "KS Test" : kstests
    })

table
```

Out[51]:

	Confidence interval	KS Test	Mean	Median	Size	Std	Type
0	(-1.0, 0.067)	(0.17,0.93)	-0.47	-0.27	10	0.86	Normal
1	(-0.2, 0.23)	(0.069,0.73)	0.017	0.0028	100	1.1	Normal
2	(-0.1, 0.025)	(0.032,0.24)	-0.039	-0.075	1000	1.0	Normal
3	(-0.028, 0.012)	(0.005,0.97)	-0.0077	-0.0031	10000	1.0	Normal
4	(1.1, 2.6)	(0.4,0.057)	1.8	1.4	10	1.2	Pareto
5	(1.3, 1.6)	(0.23,3.8e-05)	1.5	1.2	100	0.62	Pareto
6	(1.4, 1.5)	(0.24,0.0)	1.5	1.3	1000	0.68	Pareto
7	(1.5, 1.5)	(0.26,0.0)	1.5	1.3	10000	0.78	Pareto

8	(0.28, 0.58)	(0.15,0.97)	0.43	0.43	10	0.24	Uniform
9	(0.48, 0.59)	(0.082,0.5)	0.54	0.55	100	0.29	Uniform

	Confidence interval	**KS Test**	**Mean**	**Median**	**Size**	**Std**	**Type**
10	(0.48, 0.52)	(0.069,0.00012)	0.5	0.5	1000	0.29	Uniform
11	(0.5, 0.51)	(0.059,3.3e-30)	0.5	0.5	10000	0.29	Unifrom

Вывод

1) Нормальное распределение не отвергается при любом размере выборки

2) Парето(всегда) и Равномерное(при большой выборке) отвергается

3) Равномерное(при маленькой выборке) не отвергается

Т.е. значение статистики Колмогорова-Смирнова зависит от размера и распределения выборки

Скачайте данные http://archive.ics.uci.edu/ml/datasets/Wine (http://archive.ics.uci.edu/ml/datasets/Wine), файл `wine.data`. Что вы можете сказать про столбцы 1, 4, 8 (нумерация с нуля), соответсвующие 'Alcohol', 'Alcalinity of ash', 'Nonflavanoid phenols'?

In [65]:

```
import seaborn as sns
```

In [66]:

```
names = ['Alcohol', 'Malic acid', 'Ash', 'Alcalinity of ash', 'Magnesium',
         'Total phenols', 'Flavanoids', 'Nonflavanoid phenols', 'Proanthocyanins',
         'Color intensity', 'Hue', 'OD280/OD315 of diluted wines', 'Proline']
```

In [73]:

```
df = pd.read_csv('wine.data', header=None, names=names, index_col=None)
sns.pairplot(df[['Alcohol', 'Alcalinity of ash', 'Nonflavanoid phenols']], diag_kind='kde')
```

```
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\nonparametric\kdeto
ols.py:20: VisibleDeprecationWarning: using a non-integer number instead o
f an integer will result in an error in the future
  y = X[:m/2+1] + np.r_[0,X[m/2+1:],0]*1j
```

Out[73]:

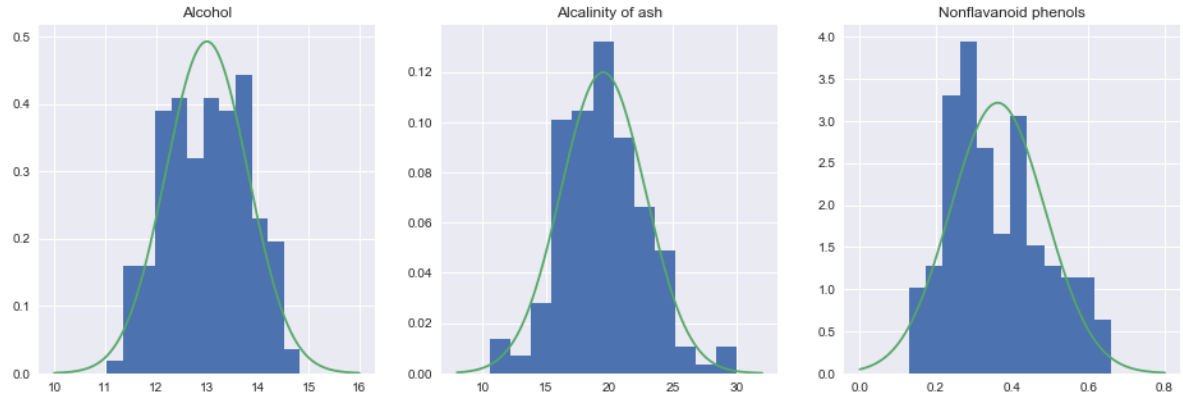
```
<seaborn.axisgrid.PairGrid at 0x1d61cb3de80>
```

Похоже на нормальное распределение. Попробуем найти его параметры

In [81]:

```
sizes, means, medians, stds, intervals, kstests = [], [], [], [], [], []

plt.figure(figsize=(16, 5))

for pos, label, start, finish in zip([1, 2, 3],
                                     ['Alcohol', 'Alcalinity of ash', 'Nonflavanoid phenols'
                                     [10, 8, 0],
                                     [16, 32, 0.8]):
    size, sample_mean, sample_median, sample_std, confidence_interval, kstest = normal_summ

    plt.subplot(1, 3, pos)
    plt.hist(df[label], normed=True, bins=12)
    plt.title(label)
    x = np.linspace(start, finish, 1000)
    plt.plot(x, sps.norm(loc=sample_mean, scale=sample_std).pdf(x))

    print(label)
    print('size: %d' % size)
    print('sample mean: %.2f' % sample_mean)
    print('sample median: %.2f' % sample_median)
    print('sample std: %.2f' % sample_std)
    print('0.95 confidence interval: (%.2f, %.2f)' % (confidence_interval[0], confidence_ir
    print('KS-stat: %.3f' % kstest.statistic)
    print()
plt.show()
```

```
Alcohol
size: 178
sample mean: 13.00
sample median: 13.05
sample std: 0.81
0.95 confidence interval: (12.88, 13.12)
KS-stat: 0.069

Alcalinity of ash
size: 178
sample mean: 19.49
sample median: 19.50
sample std: 3.33
0.95 confidence interval: (19.01, 19.98)
KS-stat: 0.063

Nonflavanoid phenols
size: 178
sample mean: 0.36
sample median: 0.34
sample std: 0.12
0.95 confidence interval: (0.34, 0.38)
KS-stat: 0.115
```



Вывод

Таким образом, эти столбцы действительно распределены нормально с указанными параметрами

2. Байесовские методы

Задача 7. Пусть $X_1, \ldots, X_n \sim \mathcal{N}(\theta, 1)$ и θ имеет априорное распределение Коши. Как было сказано на лекции, аналитически интеграл в знаменателе формулы Байеса посчитать не удастся. Однако, поскольку в данном случае параметр один, можно его посчитать с помощью приближенного интегрирования.

В качестве метода приближенного интегрирования можно использовать следующую можификацию известного метода Монте-Карло. В качестве оценки интеграла $\int\limits_{\mathbb{R}} f(x)p(x)dx$, где $p(x)$ --- некоторая плотность, можно взять величину $\sum\limits_{j=1}^{k} f(Y_i)$, где $Y_1, \ldots, Y_k$ --- сгенерированная выборка из распределения, имеющего плотность $p(x)$.

Сгенерируйте выборку размера 5 из стандартного нормального распределения. Посчитайте для нее c --- знаменатель в формуле Байеса. Какой размер вспомогательной выборки в методе приближенного интегрирования необходим, чтобы с большой точностью посчитать значение c?

Нарисуйте график плотности апостериорного распределения. Посчитайте математическое ожидание по апостериорному распределению.

Задача 8. Рассмотрим схему испытаний Бернулли (т.е. броски монет) с вероятностью успеха p.

Постройте несколько графиков сопряженного распределения для разных параметров и охарактеризуйте, как значения параметров его соотносятся с априорными знаниями о монете. Это могут быть, например, знания вида

- монета скорее честная (при таком априорном распределении наиболее вероятны значения p в окрестности 0.5)
- монета скорее нечестная, перевес неизвестен (наименее вероятны значения p в окрестности 0.5)
- монета скорее нечестная, перевес в сторону герба (наиболее вероятны значения p в окрестности 1)
- монета скорее честная, либо с небольшим перекосом вправо (наиболее вероятны значения p в окрестности ~0.6)
- ничего не известно (все значения равновероятны)

Для каждого случая из перечисленных выше постройте график плотности сопряженного распределения (на одной фигуре).

Ниже приведена реализация некоторых вспомогательных функций.

In [ ]:

```
def draw_posteriori(grid, distr_class, post_params, xlim=None):
    ''' Рисует серию графиков апостериорных плотностей.
        grid --- сетка для построения графика
        distr_class --- класс распределений из scipy.stats
        post_params --- параметры апостериорных распределений
                        shape=(размер выборки, кол-во параметров)
    '''

    size = post_params.shape[0] - 1

    plt.figure(figsize=(12, 7))
    for n in range(size+1):
        plt.plot(grid,
                 distr_class(post_params[n]).pdf(grid) if np.isscalar(post_params[n])
                 else distr_class(*post_params[n]).pdf(grid),
                 label='n={}: {}'.format(n, post_params[n]),
                 lw=2.5,
                 color=(1-n/size, n/size, 0))
    plt.grid(ls=':')
    plt.legend()
    plt.xlim(xlim)
    plt.show()


def draw_estimations(ml, distr_class, post_params, confint=True, ylim=None):
    ''' Рисует графики байесовской оценки (м.о. и дов. инт.) и ОМП.
        ml --- Оценка максимального правдоподобия для 1 <= n <= len(sample)
        distr_class --- класс распределений из scipy.stats
        post_params --- параметры апостериорных распределений
                        shape=(размер выборки+1, кол-во параметров)
    '''

    size = len(ml)
    distrs = []
    for n in range(size+1):
        distrs.append(distr_class(post_params[n]) if np.isscalar(post_params[n])
                      else distr_class(*post_params[n]))

    plt.figure(figsize=(12, 4))
    plt.plot(np.arange(size+1), [d.mean() for d in distrs], label='Bayes', lw=1.5)
    plt.fill_between(np.arange(size+1), [d.ppf(0.975) for d in distrs],
                     [d.ppf(0.025) for d in distrs], alpha=0.1)
    plt.plot(np.arange(size)+1, ml, label='ML', lw=1.5)
    plt.grid(ls=':')
    plt.ylim(ylim)
    plt.legend()
    plt.show()
```

Реализуйте следующую функцию